Chromasette Magazine
P.O. Box 1087,
Santa Barbara, CA 93102
(805) 963-1066

AUGUST 1981

That's two...

We must be official! This is our second issue! Since CLOAD Magazine (for the Models I and III) was formed 3 years ago, we have seen quite a few cassette-based magazines fail to get out issue #1 (let alone #2!!). So we will break open the champagne once again as this issue of Chromasette goes to the P.O. (it makes us feel better when we have an excuse to party at work). There has been only one complaint - Tom is discovering that he now has to WORK...

Filename	English Translation	PMODE	PCLEAR	Locations
HORNCOV	Horn Cover	3	4	8 & 137
DRAWINST	Drawer Instructions	(2)	(4)	25 & 151
DRAWER	Drawer	Ø-4	6	45 & 166
WORDS	Words	(2)	(3)	63 & 18Ø
JERUSADV	Jerusalem Adventure	Ø	1	83 & 196
LANDER	Lander	(2)	(4)	1Ø7 & 215
TWODATES	Two Dates	(2)	(4)	124 & 229 use

Locations are for the R/S CTR-8Ø. If the first copy of a program won't load, try the second. If neither copy loads, return the tape for disciplining and a prompt replacement. PMODE and PCLEAR values in parentheses are not explicitly set in the programs and may have to be entered before loading or running the programs. Otherwise, an OM, FC, or SN error may occur.

Horn Cover (named for the occasional time that the graphic resembles a horn of plenty) is a demonstration of the DRAW command. All the patterns were created by just manipulating the Scale, Color, No-update, and Angle options of the DRAW command, and sticking them inside a few FOR-NEXT loops. However, there is a little tweeking done on the starting positions of each figure so that they all share a common center.

You've been playing with your Color Computer a while, but you still don't quite understand all of the ins and outs of the graphics. Drawer allows you to really PLAY with the graphic features. You can draw a picture, paint it, change the colors, move blocks of the screen around, change the start page, change the PMODE (this one is FUN), save your drawing to tape, etc.

First, you should run Drawer Instructions. Run 'em a couple of times until you have a good idea what the Drawer program allows you to do. Does it seem that there are a lot of options? You're right! Don't be alarmed, however. If you are drawing away and you want to do one of the options, but you don't know which one or how to use it, just type 'H' (for Help). You will then get a thumbnail sketch of all of the commands and a list of all of important variables. Then you return to your drawing, armed with your next command.

That variable list can be a real lifesaver. More than a few times it appeared that the program was acting funny. So I hit 'H' and looked at the variable list. And usually the 'Last Command' variable was something besides 'H'. That meant that the program was waiting for me to finish a previous command. So I would finish up that command and go back to drawing. The variables also saved me when I wanted to Wash (PAINT) in a figure. I would it 'W', and then try to remember what the foreground and backround colors were. No problem - I just hit 'H' and looked at what the colors were in the variable list.

Special notes - if you have 32K, you can get more than the 6 graphic pages by changing the PCLEAR6 in line 10 to PCLEAR8, changing the MP=6 to MP=8 in line 12, and (for cosmetics) changing the 'P' and 'R' command references in lines 224 and 226 to go from Pages 1-8. Also, stay away from the red joystick buttons. Pressing one of them can cause you to execute a random command (and you'll have to go into the Help subroutine to see what command you're in) since the right joystick button generates @ABCDEFG and the left one generates HIJKLMNO.

More special notes - In a 16K machine, you only have a couple hundred bytes of free memory left after running Drawer. This is plenty for all of the operations... except for Wash (which uses the Extended BASIC PAINT command). If you Wash an intricate drawing, on occasion you will get an OM (Out of Memory) error. Have you noticed how the computer PAINTs? It starts painting down from the specified point, then goes back to fill in those areas it missed. Well, somehow the computer has to keep track of where it has painted to be able to go back to catch the nooks and crannies. It does this by building a 'stack' of numbers representing the boundries of places it still has to visit. As it fills an area with color, it adds to the stack when it encounters an area to be painted on a future pass and takes a value off the stack when it finishes painting an area. Evidently, the stack is stored in the free memory area, so if it gets too large, you get an OM error. There is way to save your drawing, however -

If you get an error during Wash (or any other error other than a SN error), just type 'SCREEN 1,SC : GOTO 20'<enter> right away and save your drawing to tape (with the 'T' option). It takes up to 7 minutes to save the entire screen. Then RUN Drawer again (RUNning cleans up the stack), and load your drawing back in with the 'T' command. If you accidentally hit <break>, you can get back to where you were by typing 'SCREEN 1,SC : CONT'<enter>.

Ok, Drawer sounds like fun, right? But can it really do anything? Many of you told us that you thought last month's cover program was neat. With Drawer, you can easily change that Chromasette banner to say anything you want to! First, you create your own banner following these rules:

1) Be sure you're in PMODE 3 (use the 'G' command).
2) Draw your banner between 0-255 in the X direction and between 0-41 in the Y direction. It is helpful to draw a boundary line across the screen at Y=42. You can get the cursor's X and Y values from the Help command.
3) Do not have more than 4 pixels in a row of Green or Buff. These colors are represented by binary '00', and 4 of them in a row would make a byte with the value of 0. 0 is a special marker in BASIC programs and would be disastrous if it is packed into a string.

Second, you save the block surrounding your banner to tape with the 'T' command. BE SURE that the block you save goes the full length of the screen ' goes at least from 0 to 41 in the Y direction (you can save a little more the Y direction but only the first 42 lines are used by the cover program).

Now, load in last month's First Cover and add the following code:

```
69 GOTO 30000
30000 CLS : INPUT"<ENTER> WHEN CASSETTE READY TO LOAD":O
```

```
30020 VP=0 : AD=0 : I=0 : J=0 : REM DECLARE VARS TO BE USED
30030 INPUT#-1,Q,Q,Q,Q : REM GET RID OF EXTRA VALS USED BY DRAWER
30040 FOR I = 0 TO 6 : REM TO FILL UP 7 STRINGS
30050 VP=VARPTR(LO$(I)) : REM GET STRING DESCRIPTOR
30060 AD=PEEK(VP+3)+PEEK(VP+2)*256 : REM GET STRING LOCATION
30070 FOR J = 0 TO 191 : REM DO ALL BYTES IN STRING
30080 INPUT#-1,Q : REM GET GRAPHIC BYTE
30090 POKE(AD+J),Q : REM PUT GRAPHIC BYTE IN STRING
30100 NEXTJ : NEXTI
30110 CLOSE : DEL 69 : REM STRINGS NOW PACKED - RUN COVER AGAIN
```

Run the modified cover program, load in the data you got from Drawer, then run the cover program again to see your handywork! Thankees - this idea came from the local Radio Shack Computer Center.

Words is the first in a probably long series of unscramble-the-word programs that I will see. And you may even catch more than a couple of them.

Treasures worth their weight in oil! How do you get them? Let's go on a little Jerusalem Adventure. For those of you who are unfamiliar with 'role playing' games (hee, hee, hee), you may have a couple of frustrating hours trying to get off that @#*&%@$ street in Jerusalem. You see, you are an adventurer in this unfamiliar city, and you are trying to find 9 treasures located somewhere around there. By observing your surroundings and giving the appropriate commands, you collect treasures and get hints on how to get around other obstacles in order to find more treasures.

Be forewarned! If you find yourself stuck in Jerusalem Adventure and you want help, calling here may not be a good idea. I take sadistic pleasure in not giving hints to adventure games! However, I may be a bit nicer since it is quite tricky to get off that street in Jerusalem. Then again, maybe I'll just go for greater personal pleasure...

Lander is the first in a probably long series of land-on-the-planet programs that I will see. And you may even catch more than a couple of them. Haven't you read this before?

Dave-of-a-thousand-days. If it starts August 14, 1981, it ends May 10, 1984. See what useful data you get from Two Dates? You give it the first date, then you give it a date displacement or a new date, and you get the monthly calendars for both dates as well as the number of days between the dates. You can also assign a daily or weekly value to be calculated. This program is only accurate from March 1, 1900 to February 28, 2100, so if you plan to time-travel with it, don't go too far! Or modify the program to NOT give you a leap year in years ending in '00' unless they are divisible by 400. I didn't modify the program because: a) I was lazy or, b) you needed a challenge. I like 'b' better.

Damn the interpreter, full space ahead! - This month's debugit lesson.

You have just typed a 3 line program in your pretty beast, and now it won't run correctly. You do a little detective work, and line 20 seems to be the culprit:

```
10 INPUTA,B
20 IFA>=BTHENPRINTAELSEPRINTB
30 GOTO10
```

According to the manual, it is syntactically correct!!! Now what? Well, retype line 20 and spread it out a bit:

```
20 IF A>=BTHENPRINTAELSEPRINTB : REM SPACE AFTER IF
```

That didn't help, so let's try again -

```
20 IFA>=B THENPRINTAELSEPRINTB : REM SPACE BEFORE THEN
```

Hey, we don't get a SN error! But the value printed out isn't right either. Durn it! Ok, time to hit line 20 again:

```
20 IFA>=B THEN PRINTAELSEPRINTB : REM SPACE BEFORE PRINT
```

Bugs still. Go for another round -

```
20 IFA>=B THENPRINT AELSEPRINTB : REM SPACE AFTER PRINT
```

Bah, humbug! One more try before trashing this computer:

```
20 IFA>=B THENPRINTA ELSEPRINTB : REM SPACE BEFORE ELSE
```

Yahoo! Woopie! And it only took 2 hours! Sure is strange... what if we retype the line as:

```
20 IFA>=2THENPRINT1ELSEPRINTB : REM PUT #S BEFORE THEN AND ELSE
```

Now the program doesn't do anything worthwhile, but it does work. From this exercise we can make this hypothesis (can't we?):

All keywords (those words recognizable by BASIC) must be preceded by:
1) a space,
2) a number,
3) some punctuation, or
4) another keyword.
or you get funny (?) results.

Movin' up in the world...

There are a couple of interesting things for the Color Computer coming from Radio Shack yesterday - that is, they have been announced and are in the catalog but I haven't seen them yet.

You can get another 16K for your machine! I understand that the chips will be piggy-backed somehow (wonder what happened to the ol' heat problem?).

You can get disks! Your local R/S service center has to go into your baby and hack a bit (something about adding jumpers and adding RF shielding), but you get up to 4 disks connected at the cartridge slot. A preliminary list of the DOS commands showed the Color DOS to be similar to TRSDOS. And the disk drives themselves are Shugarts with a 35 track format. Ok, where is this stuff?!!

Promises, promises....

As promised last month, on the next page you will find the source listings of the various machine language routines used in First Cover (and subsequent covers), Blockade, and Jerusalem Adventure. These routines were used mainly to assist in moving blocks of data around quickly, so they are rather simple. Also, they are relocatable (Jerusalem Adventure places data in a fixed location, however) so the origin of the routines is arbitrary.

Off to the hills,

```
;This takes[illegible]ta from the strings LOS(0-6) in cover programs
;and puts [illegible] the graphic screen memory moving left to right.
;
3000     BD B3ED    JSR  B3ED    ;Get addrs of LOS(0) from BASIC
3003     1F 01      TFR  D,X     ;  and put it in X register
3005   108E 05FF    LDY  #35FF   ;Initialize screen column pntr.
3009     86 20      LDA  #20     ;Put 32 column count
300B     34 02      PSHS A       ;  on stack
300D     A6 A0      LDA  Y+      ;Increment screen column pntr
300F     34 20      PSHS Y       ;  and put on stack
3011     34 10      PSHS X       ;Push strings pointer on stack
3013     86 07      LDA  #07     ;Push 7 strings count
3015     34 02      PSHS A       ;  on stack
3017     C6 06      LDB  #06     ;Init 6 rows per string count
3019     A6 84      LDA  0,X     ;Get byte from string and
301B     A7 A4      STA  0,Y     ;  stick it on the tube
301D     1E 01      EXG  D,X     ;Add 32 (by subtracting -32)
301F     83 FFE0    SUBD #FFE0   ;  to string and screen pntrs
3022     1E 01      EXG  D,X     ;  to go on to next row
3024     1E 02      EXG  D,Y
3026     83 FFE0    ADDD #FFE0
3029     1E 02      EXG  D,Y
302B     5A         DECB         ;Dec row per string count and
302C     26 EB      BNE  3019    ;  cont. if more rows in string
302E     1F 10      TFR  X,D     ;Add 14 (subtract -14) to strng
3030     83 FFF2    SUBD #FFF2   ;  pntr to point to next string
3033     1F 01      TFR  D,X
3035     35 02      PULS A       ;Get string count from stack
3037     4A         DECA         ;Dec and do another string if
3038     26 DB      BNE  3015    ;  more strings to do
303A     35 10      PULS X       ;Get initial string column pntr
303C     A6 80      LDA  X+      ;  and inc to next column
303E     35 20      PULS Y       ;Get initial screen column pntr
3040     35 02      PULS A       ;Get column count and
3042     4A         DECA         ;  dec - if columns not done
3043     26 C6      BNE  300B    ;  go do next column
3045     39         RTS          ;Home, James - back to BASIC
```

```
;This routine takes the banner at the top of cover and copies
;it to another location on the screen.  The inst. at 3014 is a
;NOP if the copied banner is to have the same colors as orig.
;This walks through and copies the banner backwards.
;
3000     BD B3ED    JSR  B3ED    ;Get displacement from BASIC
3003     C3 10C0    ADDD #10C0   ;  and add base (save top logo)
3006     1F 01      TFR  D,X     ;Put new address in X
3008   108E 0B60    LDY  #0B60   ;Put last logo location in Y
300C     86 2B      LDA  #2B     ;Get lines count
300E     34 02      PSHS A       ;Put lines count on stack
3010     C6 20      LDB  #20     ;Init bytes per line to 32
3012     A6 A2      LDA  -Y      ;Get logo byte and dec logo loc
3014     43         COMA         ;Flop color (or NOP if no flop)
3015     A7 82      STA  -X      ;Put at new logo (dec new loc)
3017     5A         DECB         ;Dec bytes per line and
3018     26 F8      BNE  3012    ;  do more if more left
301A     35 02      PULS A       ;Get lines count and
301C     4A         DECA         ;  dec - do another line if
301D     26 EF      BNE  300E    ;  more left
301F     39         RTS          ;Go home to BASIC
```

```
;This routine takes the block location passed from Bl[illegible]de and
;returns the X, Y, X+7, Y+7 values needed by the BASI[illegible]
;command.  It is actually 4 similar routines so just the source
;for the X and Y routines will be given here.
;
;** Y value **
3000     BD B3ED    JSR  B3ED    ;Get block location
3003     1F 01      TFR  D,X     ;  and put it away
3005     4F         CLRA         ;Zero Y count
3006     5F         CLRB
3007     5C         INCB         ;Inc Y count
3008     1E 01      EXG  D,X     ;Get block location and
300A     C3 FFE0    ADDD #FFE0   ;  subtract 32 (add -32)
300D     1E 01      EXG  D,X     ;Get Y count
300F     2C F6      BGE  3007    ;Do another if block loc >=0
3011     5A         DECB         ;Get rid of extra Y count
3012     7E B4F4    JMP  B4F4    ;Send Y to BASIC
;** X value **
3015     BD B3ED    JSR  B3ED    ;Get block location
3018     C3 FFE0    ADDD #FFE0   ;Subtract 32 (add -32) until
301B     2C FB      BGE  3018    ;  block loc <0
301D     83 FFE0    SUBD #FFE0   ;Add 32 (subtract -32)
3020     86 08      LDA  #08     ;Multiply by 8 for 8 X counts
3022     3D         MUL          ;  per byte
3023     7E B4F4    JMP  B4F4    ;Send X to BASIC
```

```
;The following routines were used to achieve the split screen
;effect used in Jerusalem Adventure.  The second routine stores
;every byte of the screen into the graphics screen area until
;it hits an '='.  The first routine takes the stored stuff and
;puts it back on the screen.
;
;** Storage to Screen **
2800     BD B3ED    JSR  B3ED    ;Get PRINT @ loc. from program
2803     8E 0600    LDX  #0600   ;Start of storage (source)
2806   108E 0400    LDY  #0400   ;Start of screen mem (dest.)
280A     FD 0800    STD  0800    ;Save PRINT @ from harm
280D     A6 80      LDA  X+      ;Get source byte and
280F     A7 A0      STA  Y+      ;  stick it on the screen
2811     FC 0800    LDD  0800    ;Get PRINT @ and
2814     83 0001    SUBD #0001   ;  decrement and
2817     26 F1      BNE  280A    ;  do it again until done
2819     39         RTS          ;Go home
;** Screen to Storage **
281A     8E 0400    LDX  #0400   ;Start of screen (source)
281D   108E 0600    LDY  #0600   ;Start of storage area (dest.)
2821     A6 84      LDA  0,X     ;Get byte to stuff from screen
2823     E6 80      LDB  X+      ;Get screen byte to test
2825     C0 7D      SUBB #7D     ;  and if byte is '='
2827     27 04      BEQ  282D    ;  stop stuffing
2829     A7 A0      STA  Y+      ;Stuff byte in storage area
282B     20 F4      BRA  2819    ;Go get another byte
282D     1E 01      EXG  D,X     ;Get last screen mem loc used
282F     83 0401    SUBD #0401   ;  and make it into a PRINT @
2832     7E B4F4    JMP  B4F4    ;  location - send it to BASIC
```